**1.Create authentication service that returns JWT**

**AuthenticationController**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.security.JwtUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.\*;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public Map<String, String> authenticate(Authentication authentication) {

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

Map<String, String> response = new HashMap<>();

response.put("token", token);

return response;

}

}

**JwtUtil.java**

package com.cognizant.springlearn.security;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.util.Date;

import java.security.Key;

@Component

public class JwtUtil {

private final String secret = "mysecretkeymysecretkeymysecretkey123456";

private final Key key = Keys.hmacShaKeyFor(secret.getBytes());

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10)) // 10 hours

.signWith(key, SignatureAlgorithm.HS256)

.compact();

}

}

**Securityconfig**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User.withUsername("user")

.password(passwordEncoder().encode("pwd"))

.roles("USER")

.build();

UserDetails admin = User.withUsername("admin")

.password(passwordEncoder().encode("pwd"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

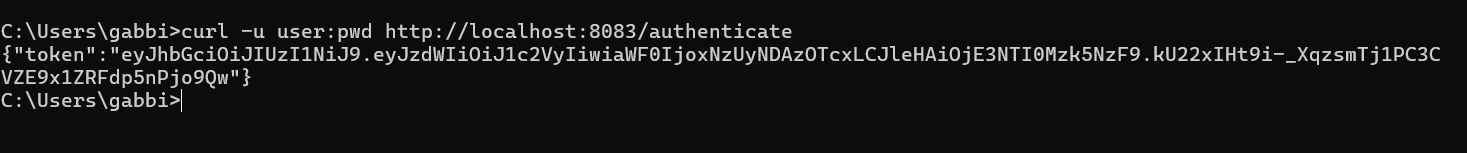
.httpBasic();

return http.build();

}

}

****

****

**2.Create authentication controller and configure it in SecurityConfig**

**AuthenticationController**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.\*;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START - /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

Map<String, String> map = new HashMap<>();

map.put("token", "");

LOGGER.info("END - /authenticate");

return map;

}

}

**SecurityConfig**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User.withUsername("user")

.password(passwordEncoder().encode("pwd"))

.roles("USER")

.build();

UserDetails admin = User.withUsername("admin")

.password(passwordEncoder().encode("pwd"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/countries").hasRole("USER")

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

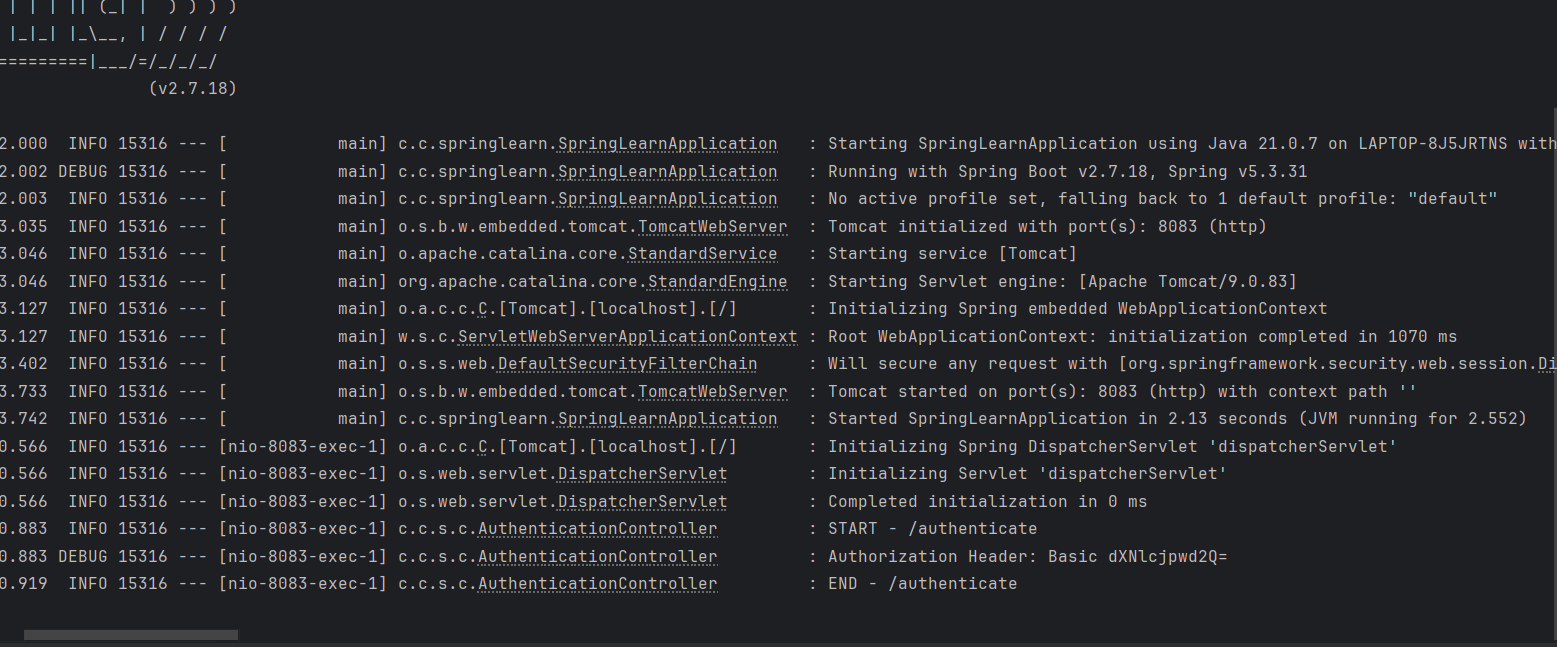
.anyRequest().authenticated()

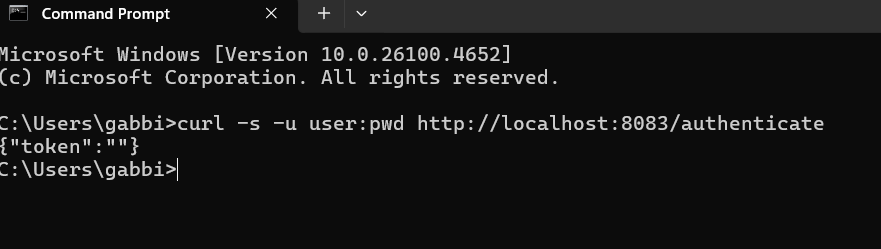
.and()

.httpBasic(); return http.build();

}

}





**3.Read Authorization header and decode the username and password**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.\*;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START - /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

LOGGER.debug("Decoded Username: {}", user);

Map<String, String> map = new HashMap<>();

map.put("token", "");

LOGGER.info("END - /authenticate");

return map;

}

private String getUser(String authHeader) {

String encodedCredentials = authHeader.substring("Basic ".length());

LOGGER.debug("Base64 Encoded Credentials: {}", encodedCredentials);

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

String decodedCredentials = new String(decodedBytes, StandardCharsets.UTF\_8);

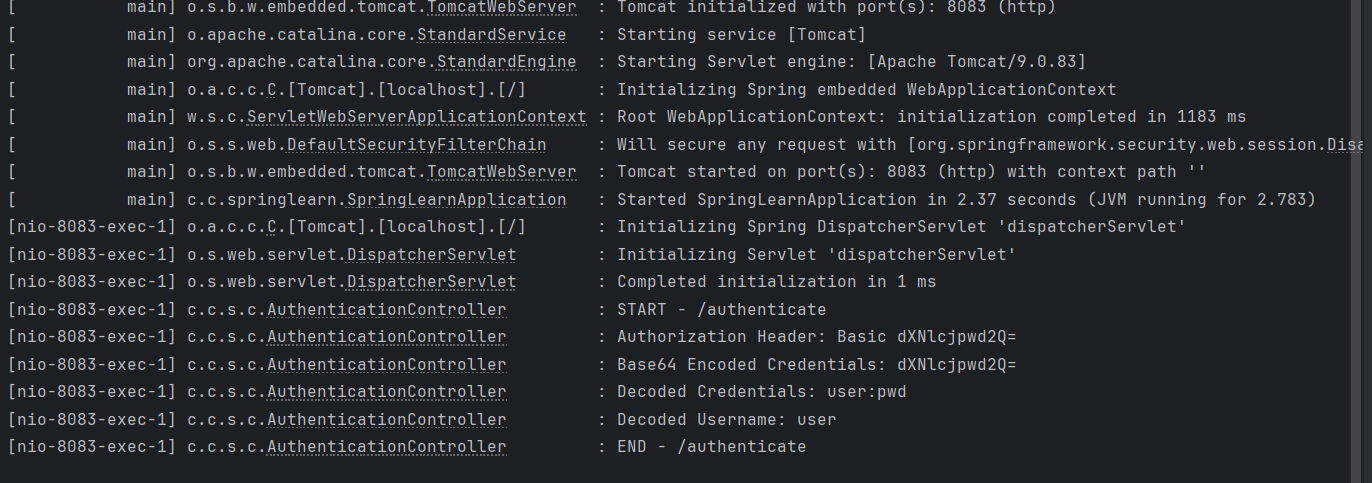
LOGGER.debug("Decoded Credentials: {}", decodedCredentials);

String username = decodedCredentials.split(":", 2)[0];

return username;

}

}



**4.Generate token based on the user**

**AuthenticationController**

package com.cognizant.springlearn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.\*;

import javax.crypto.SecretKey;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

private static final SecretKey SECRET\_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader(value = "Authorization", required = false) String authHeader) {

LOGGER.info("START - /authenticate");

Map<String, String> responseMap = new HashMap<>();

try {

if (authHeader == null || !authHeader.startsWith("Basic ")) {

LOGGER.warn("Missing or invalid Authorization header");

responseMap.put("token", "ERROR: Invalid Authorization header");

return responseMap;

}

LOGGER.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

LOGGER.debug("Decoded Username: {}", user);

String token = generateJwt(user);

LOGGER.debug("Generated JWT: {}", token);

responseMap.put("token", token);

} catch (Exception e) {

LOGGER.error("Error during authentication", e);

responseMap.put("token", "ERROR: " + e.getMessage());

}

LOGGER.info("END - /authenticate");

return responseMap;

}

private String getUser(String authHeader) {

String encodedCredentials = authHeader.substring("Basic ".length());

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

String decodedCredentials = new String(decodedBytes, StandardCharsets.UTF\_8);

return decodedCredentials.split(":", 2)[0];

}

private String generateJwt(String user) {

return Jwts.builder()

.setSubject(user)

.setIssuedAt(new Date())

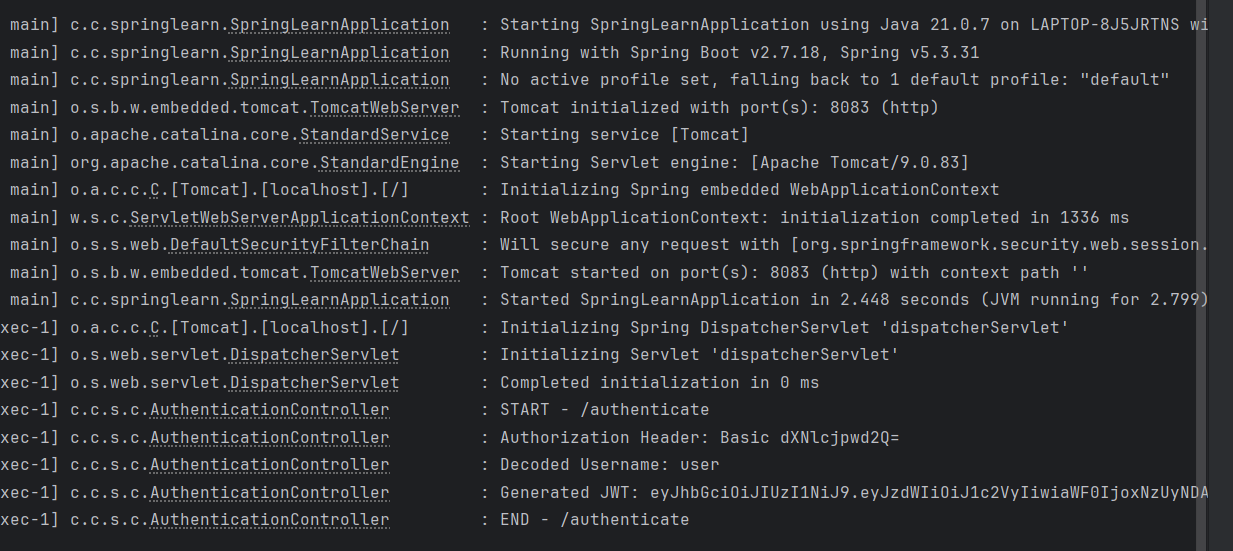
.setExpiration(new Date(System.currentTimeMillis() + 20 \* 60 \* 1000))

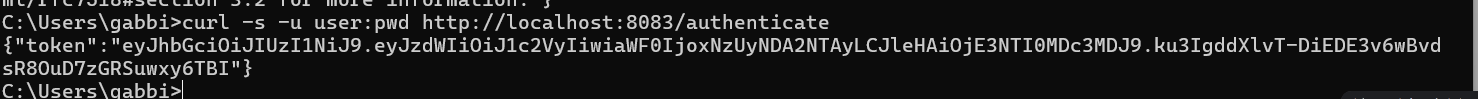
.signWith(SECRET\_KEY)

.compact();

}

}





\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5.Securing RESTful Web Services with Spring Security**

**CountryController**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.core.io.ClassPathResource;

import java.util.List;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<Country> getAllCountries() {

XmlBeanFactory factory = new XmlBeanFactory(new ClassPathResource("country.xml"));

List<Country> countryList = (List<Country>) factory.getBean("countryList");

return countryList;

}

}

**Country.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="country" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="JP"/>

<property name="name" value="Japan"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**SecurityConfig**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

@Configuration

@EnableWebSecurity

@SuppressWarnings("deprecation")

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("admin")

.password("{noop}admin123")

.roles("USER");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeRequests()

.antMatchers("/countries").authenticated()

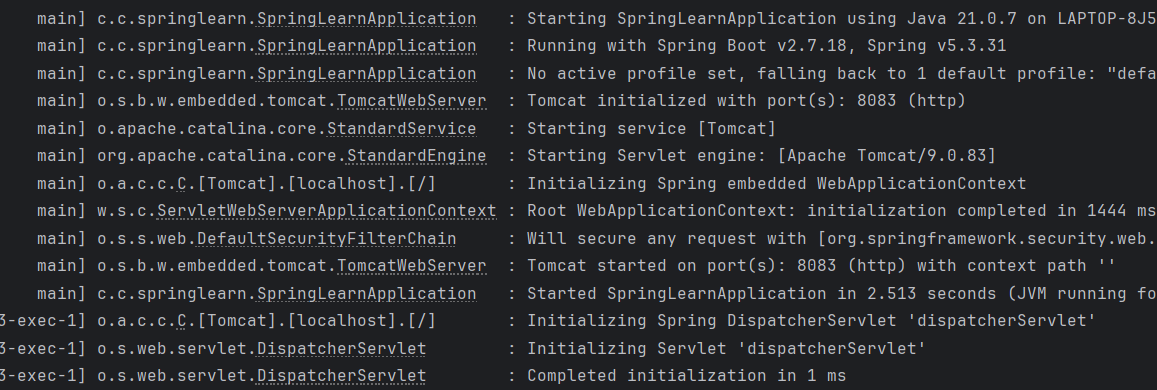
.anyRequest().permitAll()

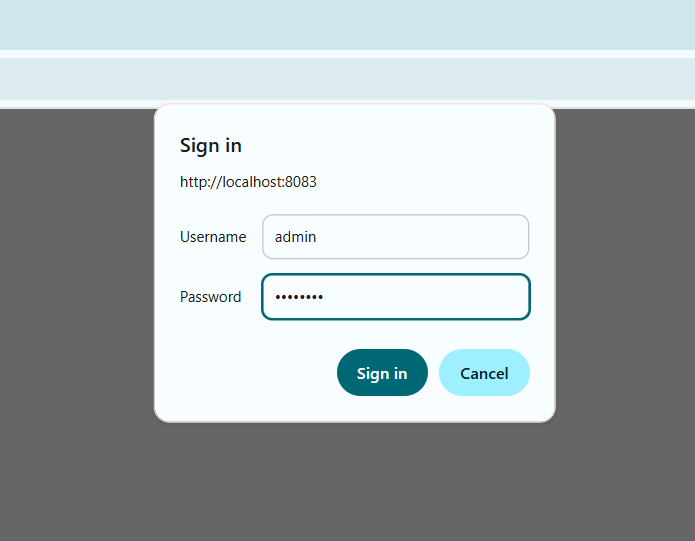
.and()

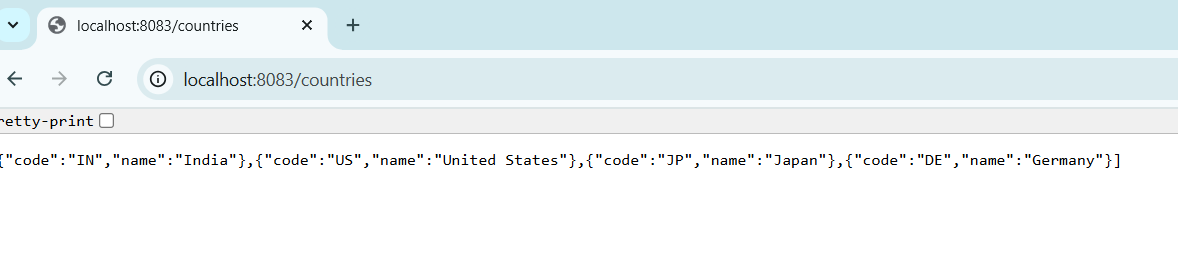
.httpBasic();

}

}







**6.Creating users and roles in Spring Security**

Securityapp.java

package com.cognizant.springlearn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Bean;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private static final Logger LOGGER = LoggerFactory.getLogger(SecurityConfig.class);

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

LOGGER.info("Start inMemoryAuthentication");

auth.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");

}

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable()

.httpBasic()

.and()

.authorizeRequests()

.antMatchers("/countries").hasRole("USER")

.anyRequest().authenticated();

}

@Bean

public PasswordEncoder passwordEncoder() {

LOGGER.info("PasswordEncoder created");

return new BCryptPasswordEncoder();

}

}

